

---

## ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

---

---

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ ΓΙΑ  
ΤΟ ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2022-2023

---

---

ΓΕΩΡΓΙΟΣ ΚΑΡΑΘΑΝΟΣ, ΑΜ:4691

ΠΑΝΑΓΙΩΤΗΣ ΚΑΛΤΣΑΣ, ΑΜ:4688

---

ΑΝΑΦΟΡΑ

ΜΑΙΟΣ 2023

## 1. ΕΙΣΑΓΩΓΗ

Στη παρούσα εργασία υλοποιήθηκε μια web εφαρμογή για αναζήτηση τραγουδιών. Στόχος ήταν η δημιουργία ενός κλώνου του Google ο οποίος θα βασίζεται σε αναζήτηση με λέξεις κλειδιά αλλά θα παρέχει και επιπλέον φιλτράρισμα στα ερωτήματα με τεχνικές που θα εξηγήσουμε στις παρακάτω ενότητες. Η μηχανή αναζήτησης που θα παρουσιάζουμε έχει υλοποιηθεί, κατά βάση, με χρήση της βιβλιοθήκης Lucene. Πέραν του παραδοσιακού τρόπου ανάκτησης πληροφορίας με ευρετηριοποίηση, που υποστηρίχθηκε από τη Lucene, έχει υλοποιηθεί και επιλογή σημασιολογικής ανάκτησης πληροφορίας και ο τρόπος με τον οποίο επιτεύχθηκε είναι το τελευταίο πράγμα που θα αναλύσουμε στην παρούσα αναφορά.

## 2. ΣΥΛΛΟΓΗ ΤΡΑΓΟΥΔΙΩΝ

Τα έγγραφα τα οποία ανακτούμε μέσα από αυτήν την εφαρμογή είναι τραγούδια. Πιο συγκεκριμένα, για κάθε τραγούδι οι πληροφορίες-πεδία που έχουμε είναι ο τίτλος του, το όνομα του καλλιτέχνη που το τραγουδάει και οι στίχοι του. Ως συλλογή τραγουδιών χρησιμοποιήσαμε ένα έτοιμο dataset από το Kaggle<sup>[1]</sup>, και βασισμένοι σε αυτό εξαγάγαμε την τελική συλλογή. Το dataset που πήραμε ήταν ένα csv αρχείο με 4 στήλες (τίτλος, καλλιτέχνης, link, στίχοι) και καθώς παρατηρήσαμε πως το link δεν είναι πραγματικό, το θεωρήσαμε ανούσια πληροφορία οπότε φιλτράραμε αυτά τα δεδομένα αφαιρώντας την σχετική στήλη. Έτσι, προέκυψε ένα τελικό csv αρχείο το οποίο έχει τις 3 εναπομένουσες στήλες και το οποίο χρησιμοποιήθηκε ως είσοδος για την κατασκευή του ευρετηρίου.

## 3. ΚΑΤΑΣΚΕΥΗ ΕΥΡΕΤΗΡΙΟΥ

Όπως και αναφέραμε, η ανάκτηση που γίνεται σε αυτή την εφαρμογή βασίζεται στη βιβλιοθήκη Lucene. Έχοντας το csv αρχείο με τα τραγούδια διαθέσιμο υλοποιήσαμε την ευερετηριοποίησή τους με τα APIs της βιβλιοθήκης. Έχουμε έγγραφα τα οποία αποτελούνται από 3 πεδία-αλφαριθμητικά και κατά την αποθήκευσή τους χρησιμοποιήθηκε η standard ανάλυση της Lucene (Standard Analyzer) η οποία αναλαμβάνει να κάνει αυτόματα πολλές βοηθητικές διαδικασίες (απαλοιφή uppercases, καθαρίζει από stop words κτλ.). Είναι σημαντικό να επισημανθεί πως η ρουτίνα που δημιουργεί το ευρετήριο κατέστηκε ανεξάρτητα και αποθήκευσε το ευρετήριο στον δίσκο, οπότε **όταν τρέχει η web εφαρμογή το ευρετήριο υπάρχει ήδη** (από τεχνικής άποψης είναι μια συνάρτηση του πηγαίου κώδικα η οποία δεν μπορεί να καλεστεί με κάποιο API από τον χρήστη αλλά ο προγραμματιστής μπορεί να την εκτελέσει για να δημιουργήσει το ευρετήριο όσο δεν έχει online την εφαρμογή).

Παρακάτω περιγράφεται η διαδικασία δημιουργίας του ευρετηρίου με ψευδοκώδικα:

```
createIndex (csv file) {
```

1. *Open an index writer.*
2. *Initialize the index to a specific path.*
3. *for each line of the csv file:*
  - a. *currentTitle = get the title of the current line/song*
  - b. *currentArtist = get the artist of the current line/song*
  - c. *currentLyrics = get the lyrics of the current line/song*
  - d. *add new Document to the index with 3 fields (currentTitle, currentArtist, currentLyrics)*
4. *Close the index writer.*

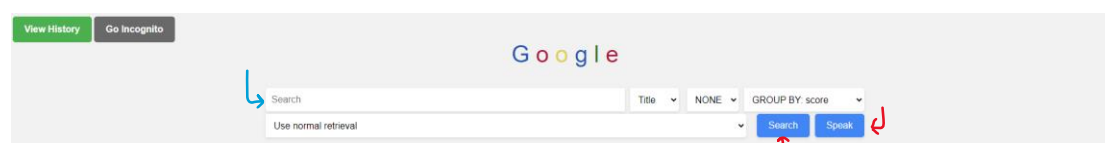
```
}
```

Για την παραπάνω διαδικασία χρησιμοποιήθηκαν τα εξής API της Lucene: StandardAnalyzer, IndexWriterConfig, IndexWriter, Document, FSDirectory και Directory.

#### 4. ΑΝΑΖΗΤΗΣΗ ΤΩΝ ΤΡΑΓΟΥΔΙΩΝ

Έχοντας το ευρετήριο έτοιμο περνάμε στην διαδικασία υλοποίησης μιας ρουτίνας που για ένα δοθέν ερώτημα κάνει αναζήτηση των 'καλύτερων' τραγουδιών. Αρχικά, πρέπει να ορίσουμε τον τρόπο με τον οποίο ο χρήστης μπορεί να κάνει τα ερωτήματα και σε δεύτερο στάδιο να πούμε πως γίνεται η αναζήτηση.

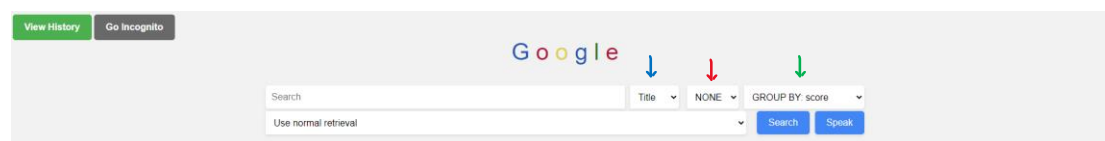
Η βασική λειτουργία της εφαρμογής είναι πως ο χρήστης μπορεί να δώσει λέξεις κλειδιά και να αναζητήσει με αυτές. Στην παρακάτω εικόνα φαίνεται με ποιο τρόπο ο χρήστης: (1) Μπορεί να πληκτρολογήσει λέξη κλειδί σε είσοδο κειμένου όπως στο κανονικό Google και (2) Να πατήσει κουμπί ηχογράφησης και να πει την λέξη κλειδί που θέλει και το σύστημα θα την βάλει σαν τιμή στην είσοδο κειμένου. Σε κάθε περίπτωση στο τέλος πατάει το κουμπί 'Search' για να γίνει η αναζήτηση.



Εικόνα 1: Είσοδοι λέξεων κλειδιών στο κλώνο του Google.

Ο χρήστης μπορεί να επιλέξει σε ποιο πεδίο από τα 3 θα πρέπει να βρίσκεται η λέξη κλειδί που εισήγαγε μέσω ενός drop down menu (1), όπως και να ψάξει με περισσότερες από μία λέξεις κλειδιά. Ο τρόπος με τον οποίο γίνεται η αναζήτηση με πολλές λέξεις κλειδιά είναι βάζοντας το '@@' ανάμεσα τους και διαλέγοντας από το σχετικό drop down menu (2) με ποιον λογικό τελεστή θα συνδυαστούν αυτές. Τέλος, ο χρήστης μπορεί να διαλέξει αν θέλει να δει τα αποτελέσματα ταξινομημένα ως προς την σχετικότητά τους με το ερώτημα που έθεσε, ή ομαδοποιημένα με βάση το πλήθος των λέξεων στον τίτλο τους (αυτή η επιλογή γίνεται με επίσης με drop down menu (3)).

Είναι εμφανές πως υπάρχει και ένα ακόμα drop down menu, αλλά προς το παρών το αφήνουμε γιατί έχει να κάνει με επιλογή από τον χρήστη για σημασιολογική αναζήτηση.



Εικόνα 2: Φίλτρα αναζήτησης.

Έχοντας λοιπόν στο λογισμικό μας σαν είσοδο το ερώτημα του χρήστη και τα flags που ερμηνεύουν την αναζήτηση την οποία επιθυμεί η ρουτίνα που υλοποιεί την εύρεση των σχετικών τραγουδιών περιγράφεται παρακάτω:

*searchSong (query, field, boolean operator, group flag) {*

1. *Open the index.*
2. *Initialize a list that will keep the results/relative songs.*
3. *Create a query object with a type depending on the Boolean operator (for example logical and-query).*
4. *Determine the max number of songs to retrieve and get the top hits-documents.*
5. *for each document in top hits:*
  - a. *current song = convert the current document to song.*
  - b. *Add the current song to the list of results.*
6. *Group the songs by rate or by title length depending on the user's option.*
7. *Return the final list of the songs.*

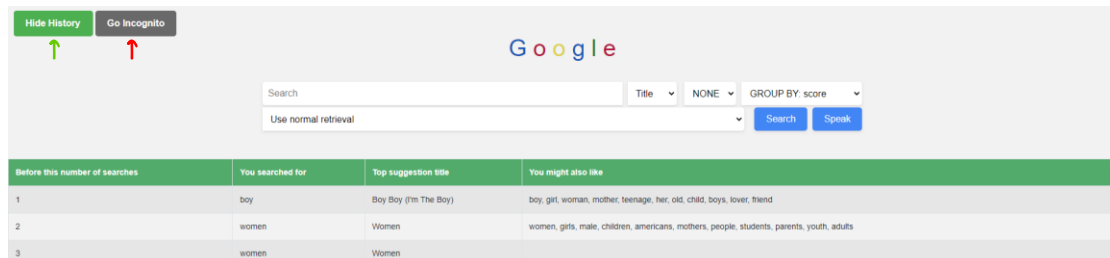
*}*

Για την παραπάνω διαδικασία χρησιμοποιήθηκαν τα εξής API της Lucene: StandardAnalyzer, Query, BooleanQuery, TopDocs, ScoreDoc, Document DirectoryReader και Directory.

Παραπάνω είδαμε με ποιον τρόπο παίρνουμε τα τραγούδια που ταιριάζουν με μια ερώτηση και στην επόμενη ενότητα θα δούμε πως τα παρουσιάζουμε, όμως υπάρχει και μια ακόμη διαδικασία που εκτελείται στη φάση της αναζήτησης και αυτή είναι η καταγραφή της. Όταν ο χρήστης κάνει μια αναζήτηση δημιουργείται μια νέα προγραμματιστική οντότητα που αντιπροσωπεύει αυτή την αναζήτηση και έχει την εξής πληροφορία: Το πόσο παλιά έκανε ο χρήστης αυτήν την ερώτηση, την ερώτηση αυτή καθαυτή, το καλύτερο τραγούδι που βρέθηκε για τη συγκεκριμένη ερώτηση και σε περίπτωση που ο χρήστης έχει επιλέξει σημασιολογική αναζήτηση κρατιούνται και σημασιολογικά κοντινές προτάσεις για μελλοντική αναζήτηση. Για τις ανάγκες αυτής της καταγραφής έχουμε δημιουργήσει έναν πίνακα 'search' σε μια βάση δεδομένων με κολόνες τα 4 πεδία που αναφέρθηκαν και οι εγγραφές αυτού του πίνακα αποτελούν το ιστορικό αναζήτησης. Η εισαγωγή νέας οντότητας στη βάση γίνεται με τη πολιτική FIFO, άρα ο πίνακας είναι μια ουρά στην οποία έχουμε θεσπίσει μέγιστο μήκος 20 (αυτό το μήκος είναι υποκειμενικό και αλλάζει πολύ ευκολά, απλά θεωρήσαμε πως δεν έχει νόημα να κρατάμε ιστορικό πριν από 20 αναζητήσεις καθώς είναι παλιό). Ο λόγος που εμπλέξαμε βάση δεδομένων στην καταγραφή είναι γιατί θέλουμε αποθήκευση του ιστορικού στον δίσκο και όχι στη μνήμη έτσι ώστε αυτό να μπορεί να ανακτηθεί και αφού επανεκκινήσουμε την εφαρμογή.

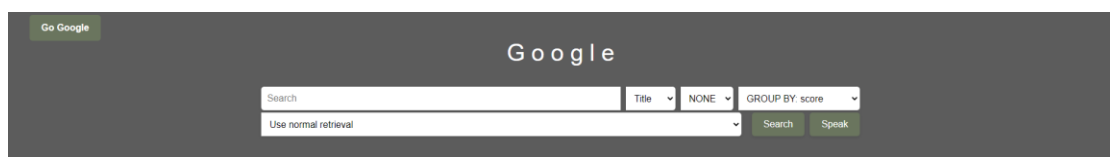
Έχοντας διαθέσιμο αυτό τον πίνακα στη βάση δεδομένων που συνδέσαμε με την εφαρμογή, προσθέσαμε ένα κουμπί (1) με το οποίο ο χρήστης μπορεί να δει το ιστορικό αλλά και ένα άλλο κουμπί (2) με το οποίο ο χρήστης μπορεί να μεταφερθεί σε σελίδα ανώνυμης περιήγησης, και οτιδήποτε ψάξει δεν θα καταγραφεί (όπως συμβαίνει και στο κανονικό Google). Στη παρακάτω εικόνα φαίνεται το αποτέλεσμα όταν έχουμε

πατήσει το κουμπί εμφάνισης ιστορικού και αυτό εμφανίζεται κάτω από τις εισόδους αναζήτησης. Στη πρώτη κολόνα είναι ένας αριθμός που δηλώνει το πόσο πρόσφατη είναι η αναζήτηση, στη δεύτερη η αναζήτηση που έγινε, στη τρίτη γίνεται μια πρόταση στον χρήστη με τίτλο τραγουδιού βάση του ερωτήματος και στη τέταρτη γίνεται μια πρόταση στον χρήστη με σημασιολογικά κοντινές λέξεις.



Before this number of searches			
You searched for			
1	boy	Boy Boy (I'm The Boy)	boy, girl, woman, mother, teenage, her, old, child, boys, lover, friend
2	women	Women	women, girls, male, children, americans, mothers, people, students, parents, youth, adults
3	women	Women	

Εικόνα 3: Ιστορικό αναζήτησης για τις 3 πιο πρόσφατες αναζητήσεις. Στη 3<sup>η</sup> περίπτωση έχουμε κάνει μη-σημασιολογική αναζήτηση για αυτό και δεν υπάρχουν σημασιολογικά εναλλακτικές προτάσεις.



Εικόνα 4: Ανώνυμη περιήγηση. Υπάρχει κουμπί μόνο για επιστροφή στο 'κανονικό Google'.

## 5. ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Έχουμε εξηγήσει πλήρως την διαδικασία της αναζήτησης, οπότε τώρα πάμε να δούμε τι γίνεται όταν ο χρήστης πατάει το κουμπί 'Search' και αναμένει να δει τα αποτελέσματα της ερώτησης του. Αρχικά να υπογραμμίσουμε το γεγονός ότι πλέον δεν χρειαζόμαστε τις παροχές της Lucene και έχοντας απλώς την λίστα που μας επέστρεψε η αναζήτηση (η οποία έχει τα τραγούδια ομαδοποιημένα είτε ανά βαθμό είτε ανά μήκος τίτλου) παρουσιάζουμε τα τραγούδια με έναν αισθητικά ωραίο τρόπο.

Μόλις ο χρήστης πατήσει το κουμπί της αναζήτησης μεταφέρεται σε μια νέα σελίδα η οποία έχει σε λίστα τα τραγούδια-αποτελέσματα. Όπως φαίνεται και στη παρακάτω εικόνα τα τραγούδια παρουσιάζονται ανά 10 και ο χρήστης μπορεί πατώντας το σχετικό κουμπί να πάει στα επόμενα ή στα προηγούμενα 10 τραγούδια. Για κάθε τραγούδι φαίνεται το όνομα του τραγουδιστή και ο τίτλος του και από δίπλα υπάρχει κουμπί για προβολή του τραγουδιού (πληροφορία που παρέχουν και πλατφόρμες σαν το Spotify).

### Songs Directory

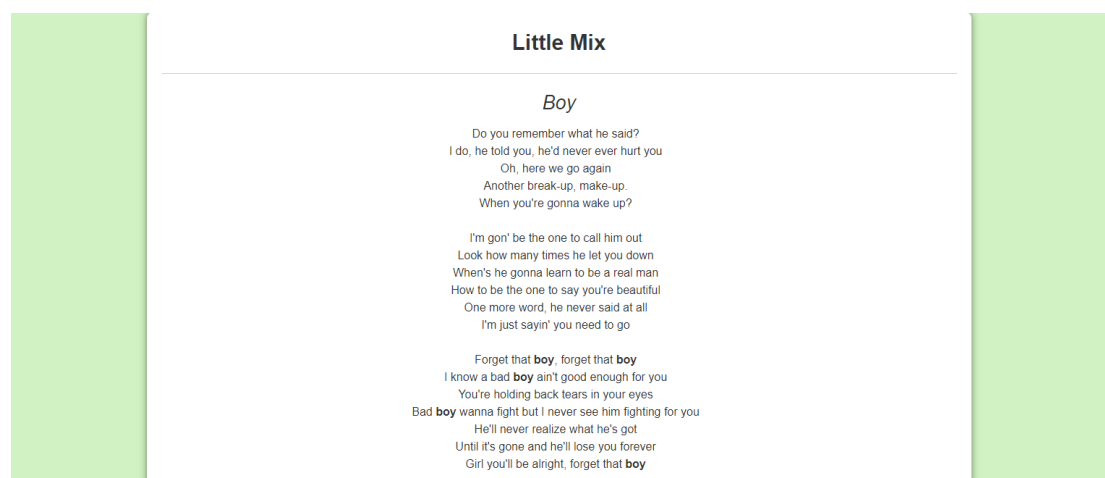
Artist's Name	Title	
Culture Club	Boy Boy (I'm The Boy)	<a href="#">View Song</a>
Little Mix	Boy	<a href="#">View Song</a>
Our Lady Peace	Boy	<a href="#">View Song</a>
Erasure	Boy	<a href="#">View Song</a>
Ian Hunter	Boy	<a href="#">View Song</a>
Kylie Minogue	Boy	<a href="#">View Song</a>
Mariah Carey	Boy	<a href="#">View Song</a>
Alabama	Dixie Boy	<a href="#">View Song</a>
Alabama	The Boy	<a href="#">View Song</a>
Alphaville	Moon Boy	<a href="#">View Song</a>

[< Show Previous](#) [Show Next 10 ->](#)

[Go back to Google Homepage](#)

Εικόνα 5: Τα τραγούδια-αποτελέσματα. Παρουσιάζονται ανά 10.

Όταν ο χρήστης πατάει το κουμπί του τραγουδιού το οποίο θέλει να δει μεταφέρεται σε μια σελίδα η οποία παρουσιάζει το τραγούδι αναλυτικά με τους στίχους του. Στους στίχους έχει γίνει highlight η λέξη κλειδί με την οποία έγινε η αναζήτηση (όπου αυτή υπάρχει).



Εικόνα 6: Η αρχή ενός τραγουδιού. Στη προκειμένη περίπτωση έχουμε αναζητήσει με τη λέξη κλειδί 'boy' για αυτό και έχει γίνει highlight όπου υπάρχει στους στίχους.

## 6. ΣΗΜΑΣΙΟΛΟΓΙΚΗ ΑΝΑΚΤΗΣΗ

Όπως αναφέραμε και προηγουμένως, η συγκεκριμένη μηχανή αναζήτησης υποστηρίζει και σημασιολογική ανάκτηση. Συγκεκριμένα, όταν ο χρήστης κάνει μια ερώτηση και έχει επιλέξει σημασιολογική ανάκτηση το σύστημα εκτελεί μια επιπλέον διαδικασία σε σχέση με πριν. Η ερώτηση του χρήστη αποτελείται από λέξεις κλειδιά και πολλές φορές όταν κάνουμε μια αναζήτηση, για παράδειγμα με την λέξη 'beach', θεωρούμε λογικό να μας εμφανισθούν αποτελέσματα παρόμοια με τη περίπτωση που θα ψάχναμε 'sun' ή 'summer' κτλ. Επομένως, στόχος την σημασιολογικής ανάκτησης σε αυτή την εφαρμογή είναι να μην βγάλει μόνο αποτελέσματα που περιέχουν την λέξη κλειδί, αλλά και αποτελέσματα που περιέχουν σημασιολογικά κοντινές λέξεις με τις λέξεις κλειδιά.

Ο τρόπος με τον οποίο το καταφέρνουμε αυτό βασίζεται στην αναπαράσταση των λέξεων με διανύσματα (word2vec). Από τη σελίδα του Stanford<sup>[2]</sup> έχουμε κατεβάσει pretrained model το οποίο έχει την διανυσματική αναπαράσταση για 6 δισεκατομμύρια λέξεις και χρησιμοποιούμε αυτό ως μοντέλο για την εύρεση των top k κοντινότερων λέξεων για μια δοθέν λέξη κλειδί. Η εύρεση γίνεται με χρήση της cosine distance και αυτό δημιουργεί ένα ερώτημα στο πως κάποιος θα προσεγγίσει να λύσει το πρόβλημα που προκύπτει με τον χρόνο εύρεσης των λέξεων. Στις δημοφιλείς μηχανές αναζήτησης λόγω του υπερσύγχρονου hardware που διαθέτουν, όσο μεγάλα και αν είναι τα μοντέλα που χρησιμοποιούν για τέτοιους σκοπούς η μνήμη που διαθέτουν οι servers τους είναι αρκετή για να διαχειριστεί τον αντίστοιχο φόρτο. Για τις ανάγκες της συγκεκριμένης εργασίας, το να φορτώνονται 6 δισεκατομμύρια διανύσματα στη μνήμη είναι κάτι που ένας απλός υπολογιστής δε μπορεί να διαχειριστεί σε απλά milliseconds οπότε αν κάνουμε σημασιολογική αναζήτηση μπορεί να πάρει και >9 δευτερόλεπτα για να δούμε το αποτέλεσμα (σε έναν πολύ ισχυρό υπολογιστή ίσως να έπαιρνε και απλά 1-

1.5 δευτερόλεπτο οπότε να μην το καταλαβαίναμε καν ότι κάνουμε σημασιολογική ανάκτηση). Δεδομένου αυτού του 'προβλήματος' ένας τρόπος για να έχουμε μια πιο γρήγορη απόκριση από το σύστημα είναι να φιλτράρουμε το pretrained model και να αφήσουμε μόνο λέξεις οι οποίες υπάρχουν και στα τραγούδια της συλλογής μας. Για παράδειγμα, δεν έχει κανένα νόημα αν ο χρήστης ψάξει την λέξη 'summer' να ανακτήσουμε από το pretrained model την λέξη 'winter' αν αυτή δεν υπάρχει σε κανένα τραγούδι και ο λόγος είναι πως ούτε θα συμβάλει στα αποτελέσματα αλλά και δεν είναι καλή πρόταση στον χρήστη ως εναλλακτικό ερώτημα αφού είμαστε σίγουροι πως δεν θα επιστρέψει κανένα τραγούδι.

## ΑΝΑΦΟΡΕΣ

- [1] Η συλλογή από το Kaggle: <https://www.kaggle.com/notshrirang/spotify-million-song-dataset>
- [2] Ιστοσελίδα του Sandford με glove: <https://nlp.stanford.edu/projects/glove/>